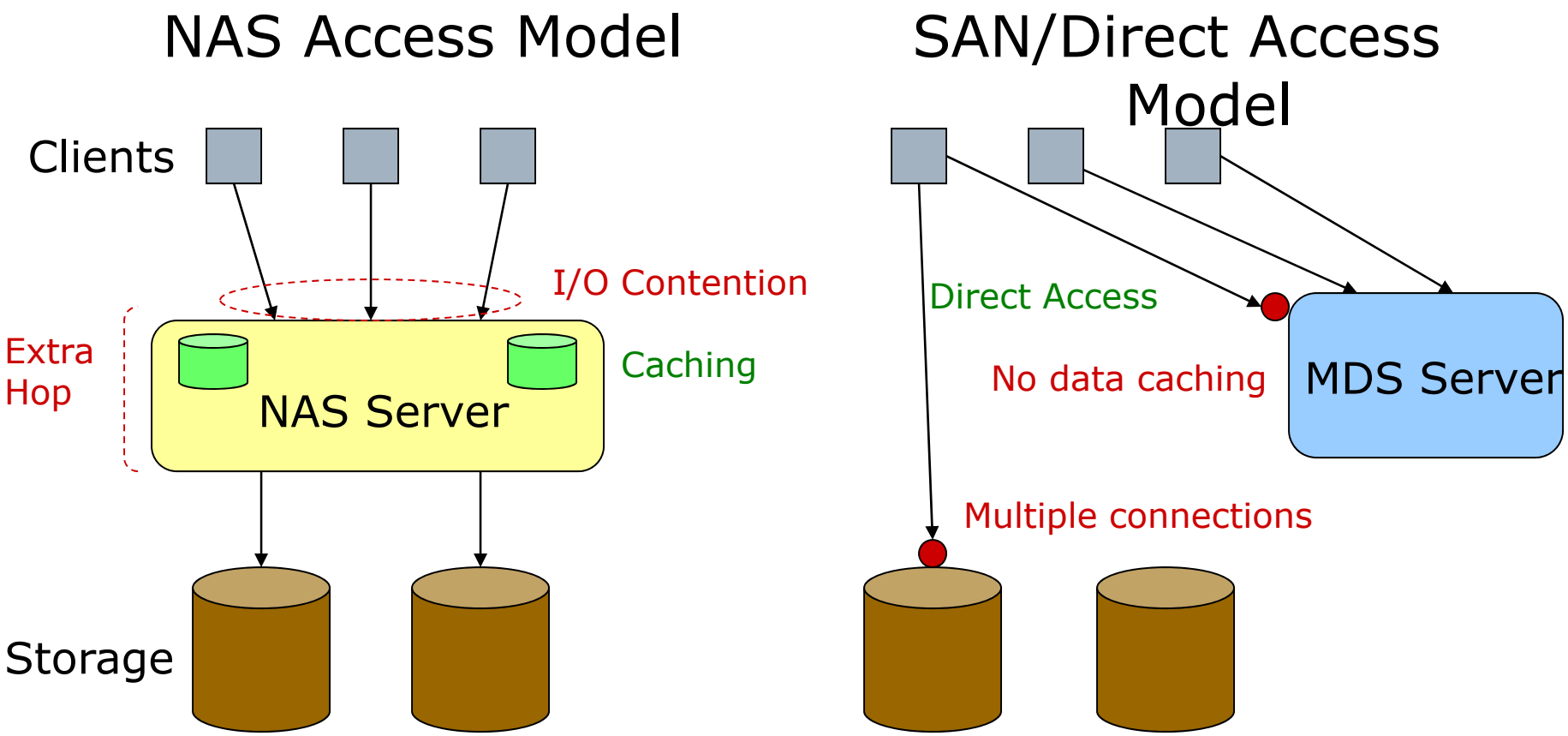
A Hybrid Access Model for Storage Area Networks

Aameek Singh Kaladhar Voruganti
Sandeep Gopisetty David Pease
Ling Liu

College of Computing
Georgia Tech
aameek@cc.gatech.edu

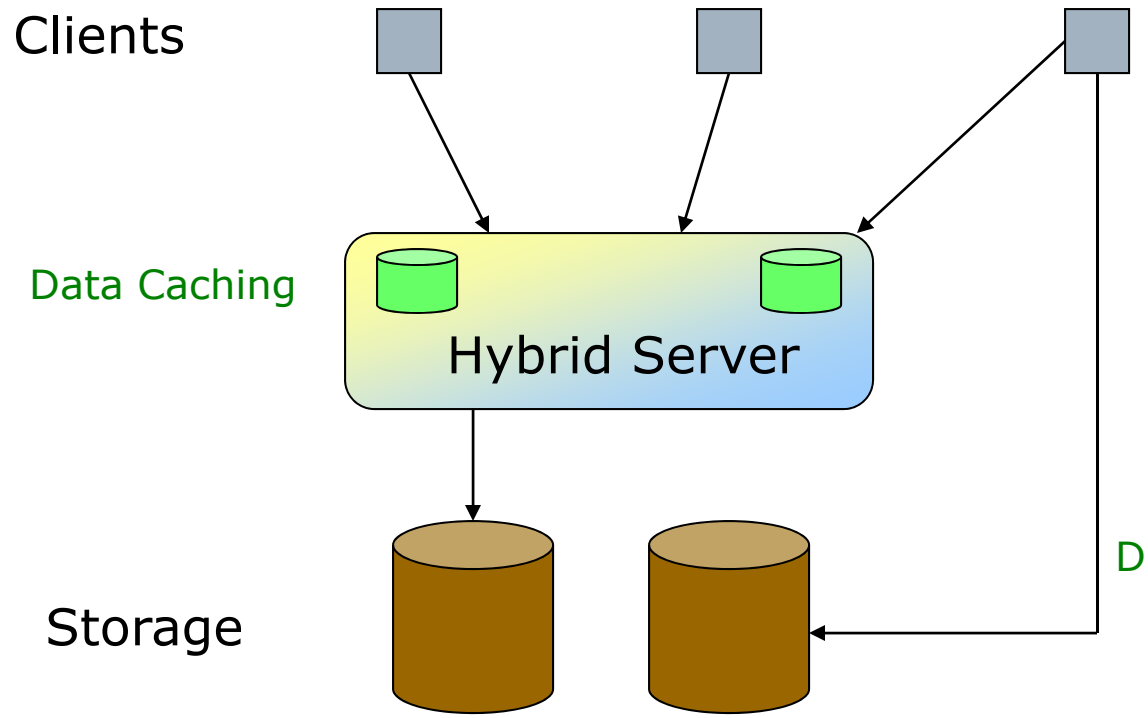
Storage Systems
IBM Almaden Research Center
kaladhar@us.ibm.com

NAS vs SAN


NAS Access Model
Clients
I/O Contention
Extra Hop
Caching
NAS Server
Storage
SAN/Direct Access Model
Direct Access
No data caching
MDS Server
Multiple connections

HSAN – Hybrid SAN

A New Hybrid Access Model



Choice of model is a **dynamic, online** decision aimed at reducing client response times

Choosing appropriate Access Model

- ≈ Utility-based Caching Problem at the Hybrid Server

Requested Object: O
Cache Admission Test: CAT
Cache Replacement Test: CRT

```
if (CAT(O)==success && CRT(O) ==success)  {
      access via NAS model  ;
      return object;
} else {
      access via direct model ;
      return metadata;
      <client accesses storage for the object>
}
```

Object Utility

$$Value\ (O) = \lambda.c\ /\ s^{\alpha}$$

- Rate of Access (λ)
  - *Greater the frequency of access, better to cache*
- Cost of obtaining object, if not in cache (c)
  - Greater the cost, better to cache
- Size of object (s)
  - Bigger the object, lesser its utility
- Size-penalty factor (α)
  - Used to favor smaller objects (like metadata), if required

Parameter Evaluation

- s – Available with MDS
- λ – MDS can compute
  - *Do NOT calculate accesses while object being held exclusively*
- c – Cost in terms of response times
  - Clients compute average access times
  - Communicate to MDS in subsequent requests
- α – Policy decision

Cache Admission Test

$$Value(O) > \max(\Pi, \min(Value(O_i)))$$

- Π = threshold parameter
  - Maintains quality of the cache
  - Dynamically computed as an average of the value of objects seen in the cache
  - $\Pi = avg(N)\{\min (Value(O_i)) \}$
  - It an be extended to incorporate Hybrid Server load

Cache Replacement Test

- Arrange all cached objects in increasing value order $\{O_1, O_2, \ldots, O_n\}$
- Let m be the minimal prefix, s.t.
  $size(O_1)+size(O_2)+\ldots+size(O_m) \geq size(O)$
- If $Value(O_m) < Value(O)$
  $evict(O_1, O_2, \ldots, O_m)$
- Ensures only less valuable objects are replaced

Data Writes

- Cache consistency Policies [Strong/Weak]
  - NDIR – *No-Dirty-Immediate-Replace*
    - Cached object is immediately invalidated whenever accessed for a *write*
    - Writes occur directly at storage (Direct Access)
  - NDNR – *No-Dirty-Never-Replace*
    - Cached object is marked irreplaceable
    - Client sends writes to the Hybrid Server which writes through to the disk immediately
  - NDCR – No-Dirty-Can-Replace
    - Cached object can be replaced
    - If the cached copy is replaced, a message is sent to the client to complete the write at the disk

Data Writes (contd…)

- DNR – Dirty-Never-Replace
  - Object is marked irreplaceable
  - Writes occur at HS, which lazily writes to disk
- DCR – Dirty-Can-Replace
  - Object can be replaced and a notification is sent to client (~ NDCR)

Name	Consistency	Benefits	Potential Drawbacks
NDIR	Strong	Simplicity	Evicting a valuable object
NDNR	Strong	Simplicity	Enforced keeping of a less valuable object
NDCR	Strong	Unbiased Caching	New connection opened during a write
DNR	Weak	Better performance (less I/O)	Implementation complexity
DCR	Weak	Less I/O and unbiased caching	Added Complexity and a new connection

Memory Model

- How to partition data and metadata caches?
  - **Partitioned**
    - Separate caches with fixed sizes
  - **Shared with strict priority to metadata**
    - A Metadata object is never replaced for a data object (~ value=∞)
  - **Shared with appropriate α**
    - Appropriate α value to favor metadata objects
    - Or hard-increment metadata values by Π

Conclusions and Future Work

- Presented a design of an intelligent SAN with hybrid access model
- Per-request granularity of choosing the appropriate model
- Initial analysis indicates promise of the approach

- Evaluation on real benchmarks
- Investigating more sophisticated caching mechanisms
  - (a) multi-tier caching (b) cooperative caching